inmos®

Module Motherboard Software User Guide

Contents

1 Introduction

The range of INMOS Module Motherboards[1] and Modules[2] allow many different configurations of modules and the connections between them to be specified without making physical changes to the boards. The configuration is performed by sending configuration data to the IMS C004 link switches[3] on the board. The MMS (Module Motherboard Software) is designed to make it easy to generate the data needed to configure a system of motherboards.

The MMS provides interactive control of a motherboard or a system of motherboards. It presents a menu-driven interface allowing the user to set up the motherboards and also to create configuration programs for use outside of the MMS.

This manual describes how to use the hardware and software description languages needed to describe the hardware system and the desired connections within that system, together with a description of the MMS (Standalone v1.00) program itself.

2 Using the MMS

2.1 Installing the MMS

To install the MMS put the MMS disk into floppy drive a: and type

```
a:install
```

This batch file creates the directory **mms2** on the Winchester and copies the contents of the disk into this directory. The archive program arc.exe, is then used to extract the data from the archive file. This directory now contains a bootable file, which contains an executable version of the MMS, together with a DOS batch file to run the MMS, and some example files for the IMS B008 and IMS B012 Motherboards.

The MMS uses the **ansi.sys** screen driver for screen handling, therefore a line like the following should be included in your **config.sys** :

```
device=ansi.sys
```

The MMS also uses a terminal description file called **ibmansi.itm**. In order for the MMS to access this file it is necessary to set up an environment variable called **ITERM**. This can be done by including the following line in your **autoexec.bat** file:

```
set ITERM=c:\mms2\ibmansi.itm
```

2.2 Getting Started

In the rest of this manual it is assumed that the motherboards in use have been set up, and that you are familiar with the user guides for them.

In order to be able to configure the links connecting the IMS C004s on the motherboards the MMS reads files, known as the "softwire" and "hardwire" files. The first of these contains a description of the connections that the user wants to make using the programmable link connections. The second contains a description of the hardware configuration of the boards being used.

The hardwire file is needed so that the MMS is able to determine what connections it is possible to make; it contains information on such things as the number of IMS C004s, number of module slots, and the connections between them. Once this description has been set up no changes will have to be made unless physical changes are made to the motherboard system. If you are using a single IMS B008 or IMS B012 there should not be any need to understand the information in the hardwire file in great detail as the supplied hardwire description files for these boards can be used without modification.

The softwire file is needed to specify both the connections from module to module and from module to edge on a motherboard. Unlike the hardwire file the softwire file will be tailored for the application being run.

You should read chapter 3 on describing softwire connections and study the example files supplied with the MMS before attempting to run the MMS or trying to set up your own softwire description. To get going initially it is probably be easiest to modify a copy of one of the example filesets provided.

2.3 Using the MMS

2.3.1 Running the MMS

To run the MMS type a line like the following at the DOS prompt:

```
mms2 softwirefile hardwirefile
```

replacing **softwirefile** and **hardwirefile** by files containing the softwire description and hardwire description respectively. The MMS will display a menu screen and prompt **key command**. At this point the user can enter any of the command codes listed on the menu, including **h** for help and **q** for quit.

2.3.2 Menu Options

The menu options available are as follows:

H — Help

Q — Quit

S — Set C004 links

C — Check source files

T — Toggle diagnostics

N — Network mapper

M — Manual command entry

L — Change link numbers

V — View source files

R — Reset subsystem

I — Initialise C004s

B — Create a bootable file

O — Create an occam table

The menu options are described in more detail in the following section.

Help

The help option allows the user to call up a help screen for each of the menu options. The help screen for the help option displays some information about the MMS, including implementation limits of number of IMS C004s, IMS T212s, slots, etc. The MMS version number is also displayed.

Quit

Return to DOS.

Set C004 Links

The set command performs the IMS C004 setting as specified in the softwire source file.

To carry out this command the MMS first reads the hardwire description, and builds up an internal representation of the motherboards. The MMS then attempts to boot the configuration pipeline with a special worm which allows commands to be sent to the IMS C004s. The MMS then reads the softwire file, and generates and sends the configuration commands to the configuration pipeline.

If errors are detected at any stage, they are reported and the command abandoned.

Check Source Files

The check source files command is essentially the same as the set command except that no attempt is made to perform the actual configuration of the boards. In this way it is possible to check a set of source files without having the corresponding hardware on-line.

Toggle Diagnostics

This toggles the diagnostic mode. In this mode any command sequences that are generated are also displayed on the screen.

Network Mapper

The network mapper command sends a simplified version of the transputer network tester[4] into the network. The currently set pipe-in link is used send the tester into the network. The mapper is currently able to detect IMS T212s, IMS T414s, IMS T800s and IMS M212s, although 6K bytes of memory is required, and therefore it will not be able to find the IMS T212s in the configuration pipeline as they have no external memory.

Manual Command Entry

The manual command option allows the user to send IMS C004 command sequences to any IMS C004 specified in the hardwire file. These sequences are of the same form as those generated automatically :

- IMS C004 id
- IMS C004 command
- any parameters required by the command

It is not possible to send the enquire command (BYTE 2) as no facility is provided for returning information from the configuration pipeline.

Change Link Numbers

The link change options allows the user to change the links which the MMS uses to communicate with the configuration pipeline and the module pipeline. The default settings are :

Link 1 - configuration pipeline
Link 2 - module pipeline

It is not possible to specify the same link for both pipelines.

View Source File

The view option allows the user to view the source of the softwire and hardwire files. It prompts for which file to view and which line number within that file to view. That line together with the preceding and following two are then displayed.

Reset Subsystem

The reset option asserts the subsystem reset on the host transputer, causing the system of motherboards to be reset. This will not cause the IMS C004 configuration to be lost.

Initialise C004s

The initialise option causes a software reset to be sent to each IMS C004 in the motherboard system. In order to do this the hardwire file is read to determine the number and whereabouts of each of the IMS C004s within the system.

Create a Bootable File

The bootable file option is similar to the set option except that the configuration commands generated are written to a file containing a program which will configure the network when it is booted from the server. The generated program expects the configuration pipeline to be connected to the root transputer via the configuration pipeline link set when the program is generated. This configuration program can be used without the MMS being present on the system. When run, the program will either print a message stating that the configuration was successful, or it will say at which stage it failed.

Create an Occam Table

The occam table option is similar to the set command except that the configuration commands generated are written to a file in the form of an occam table together with a program which controls the configuration pipeline during the network configuration. This occam table can be sent to the configuration pipeline using the extraordinary link communication procedures[5] to output the table. The table output will fail if the network configuration is not successful.

For example the following piece of occam can be used configure a network, assuming that the table generated by the MMS is contained in the file **mmstable.occ**:

```
#USE "\toolset\reinit"  -- OutputOrFail procedures
CHAN OF ANY to.t2 :     -- PLACE this at the appropriate link
BOOL  failed.to.boot :
TIMER clock :
INT   time :
VAL   fail.delay IS 3000 :
#INCLUDE "mmstable" -- mms2 configiguration table
SEQ
  clock ? time
  time := time PLUS fail.delay
  OutputOrFail.t(to.t2, Table, clock, time, failed.to.boot)
  IF
    failed.to.boot
      -- failed to configure network
    TRUE
      -- successfully configured network
```

3 Describing the Software Configuration

3.1 Introduction

The following sections will describe how to specify the soft connections required on a system of motherboards.

The syntax of both the softwire and hardwire descriptions are described in a modified Backus-Naur Form (BNF). For example,

> *edge.to.edge.line* = **EDGE** *edge.id* **TO** **EDGE** *edge.id*

This means "An *edge.to.edge.line* is the keyword **EDGE**, followed by an *edge.id*, followed by the keywords **TO EDGE**, followed by an *edge.id*".

A vertical bar (|) means "or", for example:

```
softwire.line = slot.to.slot.line
              | slot.to.edge.line
              | edge.to.edge.line
```

The written structure of the description is specified by the syntax. Each statement normally occupies a single line, and the indentation of each statement forms an intrinsic part of the syntax of the language. For example,

```
board.softwires.line = PIPE board.id
                         { softwire.line }
```

This means "A *board.softwires.line* is the keyword **PIPE** followed by a *board.id* followed by zero or more *softwire.lines*, each on a separate line, and indented two spaces further than **PIPE**". Curly brackets { and } are used to indicate the number of times a syntactic object occurs. { *object* } means, "zero or more *objects*, each on a separate line". Similarly $\{_1$ *object* } means, "one or more *objects*, each on a separate line. [*object*] means that *object* is optional.

Comments are introduced by a double dash (--), and extend to the end of the line.

Summaries of the syntax of the description languages are given in appendices A and B.

3.2 Softwire Definition

The softwire connections allow links on modules on a motherboard to be connected to other modules and edges, without requiring a direct hardwired route between the two. Instead the MMS routes the channels via the IMS C004s on the motherboard. It may not be possible to make every possible connection desired. This depends on how the IMS C004s and module slots are physically connected to each other.

A **SOFTWIRE** description has the following basic structure :

```
SOFTWIRE
  PIPE 0
     .
     . soft connections for board 0
     .
  PIPE 1
     .
     . soft connections for board 1
     .
  .
  .
  .
  PIPE n
    .
    . soft connections for board n
    .
END
```

The syntax of a softwire description is :

softwire.description = **SOFTWIRE**
{ *board.softwires.line*}
END

board.softwires.line = **PIPE** *board.id*
{ *softwire.line*}

The softwire lines are specified in three ways:

- Edge to edge connections
- Slot to edge connections
- Slot to slot connections

The syntax for softwire lines is:

softwire.line = *edge.to.edge.line*
| *slot.to.edge.line*
| *slot.to.slot.line*

An edge to edge connection simply specifies that the two edges named are to be connected together. For example,

```
EDGE 4 TO EDGE 7
```

The syntax for an edge to edge line is :

edge.to.edge.line = **EDGE** *edge.id* **TO** **EDGE** *edge.id*

A slot to edge line specifies that the edge is to be connected to the specifed link on the slot. For example,

```
SLOT 3, LINK 3 TO EDGE 6
```

The syntax for a slot to edge line is :

slot.to.edge.line = **SLOT** *slot.id, link.number* **TO** **EDGE** *edge.id*

A slot to slot line specifies a connection is to be made between a link on one module to a link on another module, for example:

```
SLOT 2, LINK 0 TO SLOT 1, LINK 0
```

specifies that link 0 of slot 2 will be softwired to link 0 of slot 1.

The slot to slot line has another form which includes a **VIA** statement. This form specifies that the connection is to be made via the two edges specifed. This form is really just a shorthand equivalent to two slot to edge lines. For example

```
SLOT 2, LINK 0 TO SLOT 12, LINK 3 VIA EDGE 3, 6
```

is equivalent to the longer form:

```
SLOT 2, LINK 0 TO EDGE 3
SLOT 12, LINK 3 TO EDGE 6
```

It is the user's responsibility to complete the connection by hardwiring the two edge connectors together. The purpose of this is to allow soft connections to be set up indirectly via edge links where the board architecture does not permit direct connection.

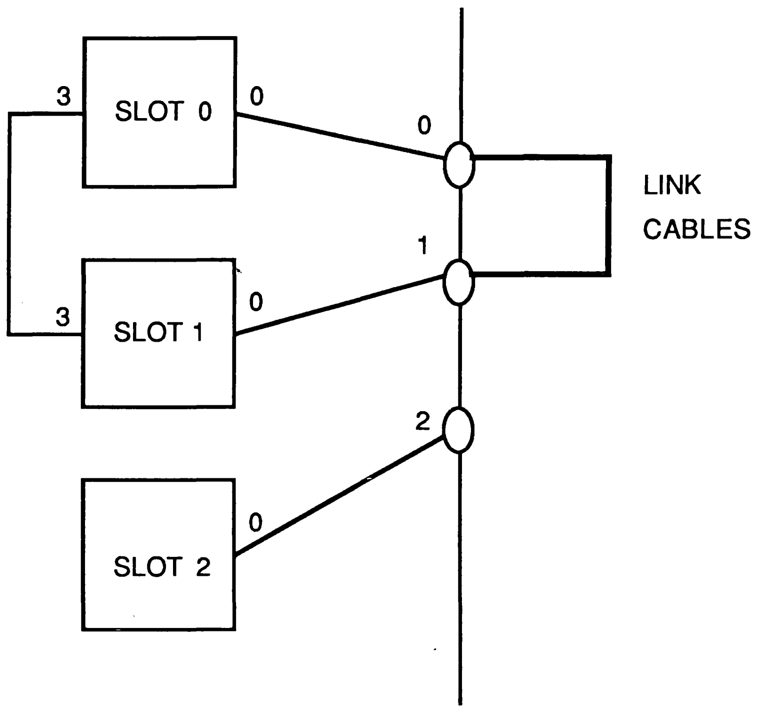
The syntax for slot to slot lines is:

slot.to.slot.line = **SLOT** *slot.id, link.number* **TO** **SLOT** *slot.id, link.number* [*via.section*]

via.section = **VIA** **EDGE** *edge.id, edge.id*

As an example of a complete file using these constructs, the following softwires file specifies all the connections in the diagram below:

```
SOFTWIRE
  PIPE 0
    SLOT 0, LINK 3 TO SLOT 1, LINK 3
    SLOT 0, LINK 0 TO SLOT 1, LINK 0 VIA EDGE 0, 1
    SLOT 2, LINK 0 TO EDGE 2
END
```


3
SLOT 0
0
0
LINK
CABLES
1
3
SLOT 1
0
2
0
SLOT 2

4 Describing the Hardware Configuration

4.1 Introduction

This chapter describes how to define the hardware configuration of a motherboard system. The MMS needs to know how the slots, IMS C004s and edges are connected together on the board in order to be able to determine whether a particular set of softwire connections is possible or not.

The following sections will describe what is required in each section of a board definition, including some examples.

4.2 Hardwire Definition

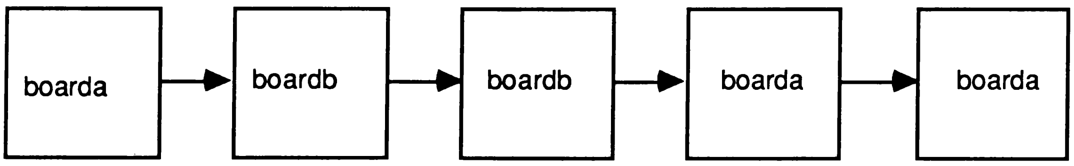
A typical hardwire definition would look something like the following:

```
.
. define boarda
.
DEF boardb
  -- sizes section
  -- t2chain section
  -- hardwires section
PIPE boarda, boardb, boardb, boarda, boarda END
```

The definition consists of two separate parts

- The definition of board types
- The definition of the pipeline

The pipeline definition tells the MMS how the boards in the system are arranged. In the example above we have the following system:



The board definition, on the other hand, specifies the connections within a particular board type. Each section of the board definition will now be described in more detail.

The syntax for a hardwire description is :

```
hardwire.description = {1 board.definition }
                        pipeline.description

board.definition     = DEF board.name
                           sizes
                           t2.chain
                           hardwires

pipeline.description = PIPE { board.name }
```

4.2.1 Sizes Section

The sizes section is used to tell the MMS how many IMS T212s, IMS C004s, slots and edges are present on the board for example:

```
SIZES
  T2   1
  C4   1
  SLOT 3
  EDGE 2
END
```

describes a board with one IMS T212, one IMS C004, three module slots, and two edge connections.

The syntax of the sizes section is :

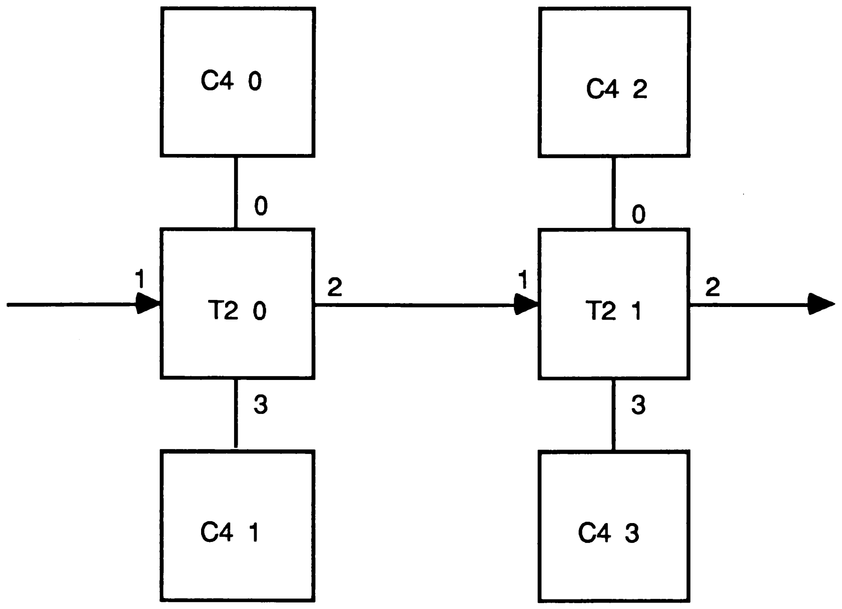
```
sizes = SIZES
          T2 positive.integer
          C4 positive.integer
          SLOT positive.integer
          EDGE positive.integer
        END
```

4.2.2 T2 Chain Section

The T2 chain section tells the MMS how the T2 chain is connected to the IMS C004s. It specifies which links of the IMS T212s are connected to the configuration links of the IMS C004s. For example,

```
T2CHAIN
  T2 0, LINK 0 C4 0
  T2 0, LINK 3 C4 1
  T2 1, LINK 0 C4 2
  T2 1, LINK 3 C4 3
END
```

describes the following system:



The syntax of the T2 chain section is:

t2.chain	= **T2CHAIN**
	{ *t2.c4.line*}
	END

t2.c4.line	= **T2** *t2.id, chain.link.number* **C4** *c4.id*
t2.id	= *positive.integer*
chain.link.number	= 0 | 3
c4.id	= 0..31

4.2.3 Hardwire Section

The hardwire section describes how the slots, edges and IMS C004s are connected together. A typical structure is as follows:

```
HARDWIRE
  --  pipeline
  --  slots to IMS C004s
  --  edges to IMS C004s
  --  slots to edges
END
```

The sections may appear in any order and lines from each may be freely intermixed, although organising it as above will aid understanding.

The syntax of the hardwire section is:

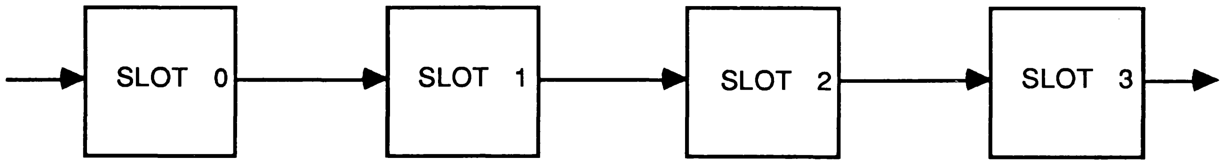
hardwires = **HARDWIRE**
  { *hardwire.line* }
  END

hardwire.line = *slot.to.slot*
  | *c4.to.slot*
  | *c4.to.edge*
  | *slot.to.edge*

The pipeline section describes how the module slots on the motherboard are connected together to form the module pipeline. In general, link 2 of a slot is connected to link 1 of the following slot so that it conforms with the module motherboard architecture[1]. It is not possible to separate the input and output channels of the links. For example,

```
SLOT 0, LINK 2 TO SLOT 1, LINK 1
SLOT 1, LINK 2 TO SLOT 2, LINK 1
SLOT 2, LINK 2 TO SLOT 3, LINK 1
```

describes the following four module pipeline



The syntax of slot to slot lines is :

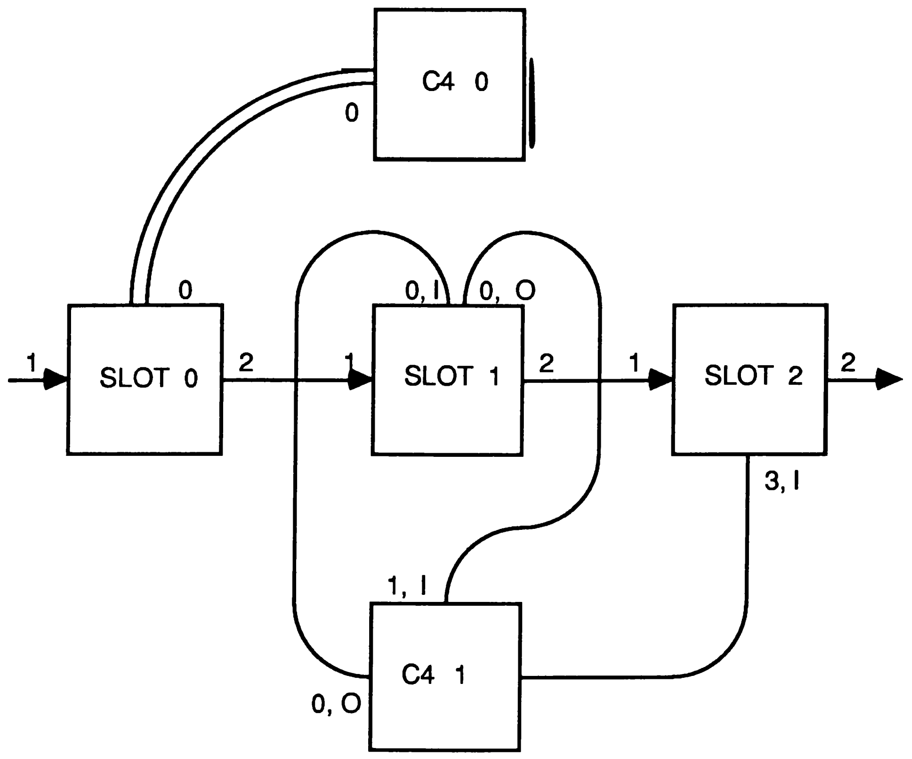
slot.to.slot = **SLOT** *slot.id, link.number* **TO** **SLOT** *slot.id, link.number*

slot.id = *positive.integer*

The slots to IMS C004s section describes how the non-pipeline links of the slots are connected to the IMS C004 link switches. In general both links 0 and 3 will be taken to an IMS C004. It is possible to specify that the input and output channels of a link are taken to different IMS C004s by including an I or O in the definition. For example,

```
C4 0, LINK 0 TO SLOT 0, LINK 0
C4 1, LINK 0, O TO SLOT 1, LINK 0, I
C4 1, LINK 1, I TO SLOT 1, LINK 0, O
C4 1, LINK 5, O TO SLOT 2, LINK 3, I
```

specifies the following connections



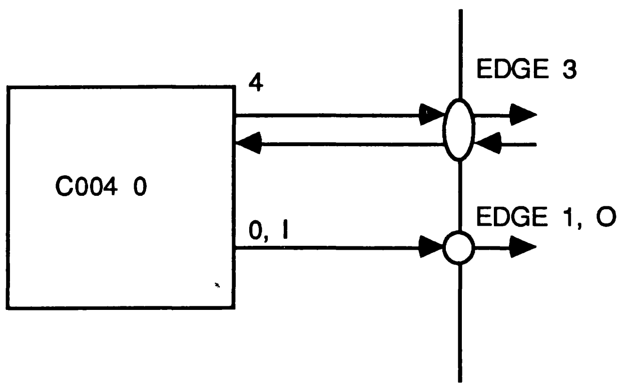
The syntax of IMS C004 to slot lines is :

c4.to.slot	= **C4** *c4.id, c4.link.no* [, *i/o*] **TO SLOT** *slot.no, link.number* [, i/o]
i/o	= **I**
	| **O**
link.number	= 0
	| 1
	| 2
	| 3
c4.link.no	= *positive.integer*

The edges to IMS C004s section specifies which edges, if any, are connected to the IMS C004s on the board. As with slots to IMS C004s, the input and output channels can be handled separately. For example,

```
C4 0, LINK 0, I TO EDGE 1, O
C4 0, LINK 4 TO EDGE 3
```

describes the following connections



The syntax of IMS C004 to edge lines is :

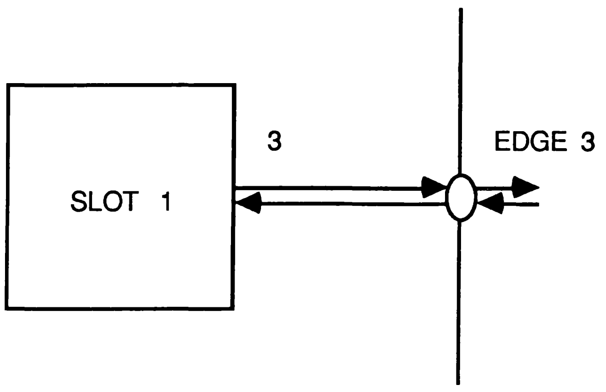
c4.to.edge = **C4** *c4.id, c4.link.no* [, *i/o*] **TO EDGE** *edge.id* [, *i/o*]

edge.id = *positive.integer*

The slots to edges section specifies which edges are connected to slots. It is not possible to separate the input and output channels for slot to edge connections. For example

```
SLOT 1, LINK 3 TO EDGE 3
```

describes the following connection:



The syntax of a slot to edge connection is:

slot.to.edge = **SLOT** *slot.id, link.number* **TO EDGE** *edge.id*

5 Error Reporting

5.1 Errors in the Hardwire Description

There are a number of different types of error that may be detected by the MMS when reading the hardwire file:

- File Reading Errors
- Syntax Errors
- Range Checking Errors
- Duplication Errors

Most error messages should be self-explanatory.

5.1.1 File Reading Errors

If the MMS is not able to read the source files an error will be reported and explained. In some cases errors of this type will be detected first as a syntax error and reported as such.

5.1.2 Syntax Errors

Any syntax errors in the hardwire file will be reported, producing one of the following types of error message:

"... unexpected symbol found ..."
"... unexpected number found ..."
"... unexpected word found ..."

The symbol that was expected at that point is usually displayed as well, together with the source line number that the error was found on. This line is also displayed in full below the error message.

For example, if the SIZES section of the hardwire file looked like this:

```
SIZES
  T2     2
  C4     4
  SLOT   32
END
```

The MMS would produce the following error message:

```
Error detected in HL1 file at line 4 :
- Unexpected symbol found ('END'). 'EDGE' was expected

Line 4 : END
```

5.1.3 Range Checking Errors

Numbers outside the following ranges will cause out of range error messages:

- implementation limit restrictions
- values defined in the SIZES section
- link values outside range 0-3

5.1.4 Duplication Errors

If any link from a slot, IMS C004 or edge is mentioned more than once in the HARDWIRE section, a duplication error will occur and an error message will be displayed. Similarly, duplicated IMS T212 links or IMS C004 IDs in the T2CHAIN section will give rise to errors.

For example,

```
.
.
C4 0, LINK 4, O TO SLOT 4, LINK 3, I
.
.
C4 0, LINK 4, O TO SLOT 7, LINK 0, I
.
.
```

will produce an error message similar to:

```
Error detected in HL1 file at line x :
- The C004 link in this connection is already involved
in a C004 to slot connection

Line x : C4 0, LINK 4, O TO SLOT 7, LINK 0, I
```

Links may not be checked for duplication in the same order as they appear in the line.

5.2 Errors in the Softwire Description

Many errors in the softwire definition are handled in the same way as the hardwire description. In addition to these errors, however, the MMS will also report soft connections which it is unable to establish. This can be for one of two reasons:

- A "hard link" mentioned in a soft connection is not defined as connected anywhere in the hardwire description
- Two hard channels are required to have a soft connection between them, but are connected to different IMS C004s making their connection impossible.

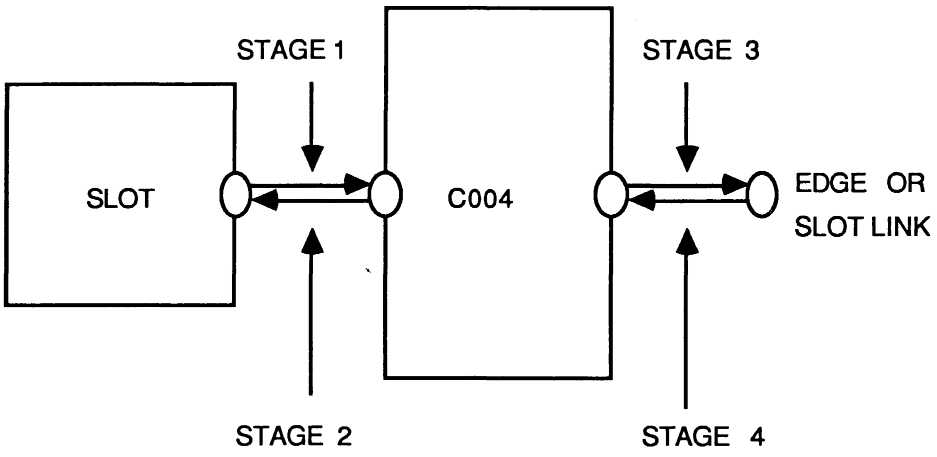
To make it easier to report and correct such errors the MMS error messages break the process of establishing a soft link down into four stages. An error may be detected and reported at any of these stages:

1 From "from link" output to IMS C004 input

2 From IMS C004 output to "to link" input

3 From "to link" output to IMS C004 input

4 From IMS C004 output to "from link" input



For example, in the following line:

```
SLOT 0, LINK 3 TO SLOT 1, LINK 0
```

the stages are as follows:

1 Check slot 0, link 3 output is connected to a IMS C004 input

2 Check IMS C004 output is connected to slot 1, link 0 input

3 Check slot 1, link 0 is connected to a IMS C004 input

4 Check IMS C004 output is connected to slot 0, link 3 input.

A Softwire Description Language

```
softwire.description = SOFTWIRE
                          { board.softwires}
                       END

board.softwires      = PIPE board.id
                          { softwire.line}

softwire.line        = slot.to.slot.line
                     | slot.to.edge.line
                     | edge.to.edge.line

slot.to.slot.line    = SLOT slot.id, link.number TO SLOT slot.id, link.number [ via.section ]

via.section          = VIA EDGE edge.id, edge.id

slot.to.edge.line    = SLOT slot.id, link.number TO EDGE edge.id

edge.to.edge.line    = EDGE edge.id TO EDGE it edges.id

link.number          = 0
                     | 1
                     | 2
                     | 3

slot.id              = positive.integer

edge.id              = positive.integer
```

B Hardwire Description Language

```
hardwire.description   = {1 board.definition }
                           pipeline.description

positive.integer       = a positive integer varying between implementations

pipeline.description   = PIPE { board.name }

board.definition       = DEF board.name
                             sizes
                             t2.chain
                             hardwires

sizes                  = SIZES
                             T2 positive.integer
                             C4 positive.integer
                             SLOT positive.integer
                             EDGE positive.integer
                           END

t2.chain               = T2CHAIN
                             { t2.c4.line}
                           END

t2.c4.line             = T2 t2.id, chain.link.number C4 c4.id

t2.id                  = positive.integer

chain.link.number      = 0
                       | 3

link.number            = 0
                       | 1
                       | 2
                       | 3

c4.id                  = 0..31
hardwires              = HARDWIRE
                             { hardwire.line }
                           END

hardwire.line          = slot.to.slot
                       | c4.to.slot
                       | c4.to.edge
                       | slot.to.edge

slot.id                = positive.integer

c4.link.no             = positive.integer

edge.id                = positive.integer

slot.to.slot           = SLOT slot.id, link.number TO  SLOT slot.id, link.number

c4.to.slot             = C4 c4.id, c4.link.no [, i/o ] TO  SLOT slot.no, link.number [, i/o ]

i/o                    = I
                       | O
```

c4.to.edge	= **C4** *c4.id, c4.link.no* [, *i/o*] **TO EDGE** *edge.id* [, *i/o*]
slot.to.edge	= **SLOT** *slot.id, link.number* **TO EDGE** *edge.id*

C The IMS C004 Programmable Link Switch

The IMS C004 programmable link switch provides a full crossbar switch between 32 link inputs and 32 link outputs. It will switch links running at standard transputer speeds (10 and 20 Mbits/sec). The IMS C004 is programmed via a separate serial link called the configuration link.

Each input and output is identified by a number in the range 0 to 31. A configuration message consisting of one, two or three bytes is transmitted on the configuration link. The configuration messages sent to the switch are shown below.

Configuration Message	Function
[0][input][output]	Connects **input** to **output**
[1] [link1] [link2]	Connects **link1** to **link2** by connecting the input of **link1** to the output of **link2** and the input of **link2** to the output of **link1**.
[2] [output]	Enquires which input the **output** is connected to. The IMS C004 responds with the input. The most significant bit of this byte indicates whether the output is connected (bit set high) or disconnected (bit set low). Early versions do not respond to the command.
[3]	This command byte must be sent at the end of every configuration sequence which sets up a connection. The IMS C004 is then ready to accept data on the connected inputs.
[4]	Resets the switch. All outputs are disconnected and held low. This also happens when **Reset** is applied to the IMS C004.
[5] [output]	Output **output** is disconnected and held low.
[6] [link1] [link2]	Disconnects the output of **link1** and the output of **link2**

For more detailed information on the IMS C004 see [3] and [6].

D The Stages of IMS C004 Configuration

This appendix is designed to give some extra information about the method used to configure the system of motherboards. The configuration takes place in a number of stages, as described below.

A special IMS T212 worm is sent down the configuration pipeline which looks for IMS T212s attached to link 2. The total number of IMS T212s found is passed back up the pipeline to the host transputer. If the number found is different from the number described in the hardwire file then an error is reported and the configuration abandoned.

At this stage a hardware reset of the IMS C004s is performed (if available on the motherboard in use), by writing to the external memory interface of the IMS T212.

The host transputer now sends the identification numbers of the IMS C004s in the system down the pipeline. This enables the worm on any particular IMS T212 to intercept commands intended for the IMS C004s it controls and pass on commands for others.

The configuration pipeline is now in a state where it is able send configuaration data to the IMS C004s.

Before sending any configuration data to the pipeline, the host transputer sends a software reset command to each IMS C004 in the system to ensure that the IMS C004s are in a known state.

The configuration data for the network is then send down the configuration pipeline, each command preceded by the identification number of the IMS C004 it is meant for.

The configuration will now be complete and it is possible to reset the system of motherboards without destroying the soft configuration.

E References

[1] "Module Motherboard Architecture", Trevor Watson, INMOS Technical Note 49, INMOS Limited, Bristol.

[2] "Dual Inline Transputer Modules (TRAMs)", Paul Walker, INMOS Technical Note 29, INMOS Limited, Bristol.

[3] "IMS C004 Programmable Link Switch", INMOS Data Sheet, INMOS Limited, Bristol.

[4] "The Transputer Network Tester", Neil Miller, INMOS Limited, Bristol.

[5] "Extraordinary Use of Transputer Links", Roger Shepherd, INMOS Technical Note 1, INMOS Limited, Bristol.

[6] "Design and Applications for the IMS C004", Glenn Hill, Technical Note 19, INMOS Limited, Bristol.

inmos®

INMOS Limited
1000 Aztec West
Almondsbury
Bristol BS12 4SQ
UK
Telephone (0454) 616616
Telex 444723

INMOS Corporation
PO Box 16000
Colorado Springs
CO 80935
USA
Telephone (303) 630 4000
TWX 910 920 4904

INMOS GmbH
Danziger Strasse 2
8057 Eching
Munich
West Germany
Telephone (089) 319 10 28
Telex 522645

INMOS Japan K.K.
4th Floor No 1 Kowa Bldg
11 - 41 Akasaka 1-chome
Minato-ku
Tokyo 107
Japan
Telephone 03-505-2840
Telex J29507 TEI JN
Fax 03-505 2844

INMOS SARL
Immeuble Monaco
7 rue Le Corbusier
SILIC 219
94518 Rungis Cedex
France
Telephone (1) 46.87.22.01
Telex 201222

June 1988
72 TDS 153 00